Designing Instagram

Let's design a photo-sharing service like Instagram, where users can upload photos to share them with other users

Instagram is a social networking service which enables its users to upload and share their photos and videos with other users. Instagram users can choose to share information either publicly or privately. Anything shared publicly can be seen by any other user, whereas privately shared content can only be accessed by a specified set of people. Instagram also enables its users to share through many other social networking platforms, such as Facebook, Twitter, Flickr, and Tumblr.

For the sake of this exercise, we plan to design a simpler version of Instagram, where a user can share photos and can also follow other users. The 'News Feed' for each user will consist of top photos of all the people the user follows.

We'll focus on the following set of requirements while designing the Instagram:

Functional Requirements:

1. Users should be able to upload/download/view photos.
2. Users can perform searches for other users based on photo titles, tags, ratings and so on.
3. Users can follow other users.
4. The system should be able to generate and display a user's News Feed consisting of top photos from all the people the user follows.
5. The system should allow adding tags to photos.
6. The system should allow searching photos on tags, ratings, dates or publishing users.
7. The system should allow commenting on photos.
8. The system should allow tagging users to photos.
9. Our services should be able to record stats of photos, e.g., likes/dislikes, total number of views, etc.
10. System should be able to list the most popular photos and users.

Non-functional Requirements:

1. Our service needs to be highly available.
2. Consistency can take a hit (in the interest of availability), if a user doesn't see a photo for a while; it should be fine.
3. The system should be highly reliable; any uploaded photo or video should never be lost.

ER Design:

In the first phase of your project, you will do the requirement analysis using ER-model. All the requirements can be obtained from the functional requirements section of this document. You must decide what kind of storage you want to use (SQL vs NoSQL) for the different entities in your system. After this phase you should generate and ER-diagram with any other supporting documentation, describing the assumptions you made. For the ER-diagram you can use any graphical editor you want, and you should finally create a PDF file using ER notations from the lectures/lab/book. You can make reasonable assumptions on your design, as long as:

- That you state them clearly in the documentation for this phase.
- They do not contradict the system requirements analysis we provide.

Schema Design:

Your task in this phase will be to translate the ER design that you made in the first step to relational database schema and determine how data will be stored in distributed file system. This includes File format, Compression and Data storage system. The relational database schema will be in a form of a single executable SQL script (*.sql file with SQL statements).

Implementation:

Your tasks in this phase will be to develop either:

- A standalone client application. This type of software development involves creation of desktop apps like Word, Excel, and Photoshop that run in the classic desktop environment and take full advantage of that environment's specific features.
- A service-based solution with a web app. This type of applications can be accessed through the Internet (or through an Intranet) using a web browser with an active internet connection. These applications are programmed using a client–server modeled structure—the user ("client") is provided services through an off-site server that is hosted by a third-party. Examples of commonly used web applications include webmail, online retail sales, online banking, and online auctions.

Because of the complexities of the service-based solutions projects that are implemented this way will receive extra bonus during the grading. Also, teams that have more than 3 members are required to implement their app as a service-based solution.

Documentation:

Product documentation describes the product that is being developed and provides instructions on how to perform various tasks with it. In general, product documentation includes requirements, tech specifications, business logic, and manuals. There are two main types of product documentation:

- System documentation represents documents that describe the system itself and its parts. It includes requirements documents, design decisions, architecture descriptions, program source code, and FAQs.
- User documentation covers manuals that are mainly prepared for end-users of the product and system administrators. User documentation includes tutorials, user guides, troubleshooting manuals, installation, and reference manuals.

Please make sure you take care of the documentation during the project development.